Scientific Paper / Artículo Científico

UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR
INGENIUS
Revista de Ciencia y Tecnología
https://doi.org/10.17163/ings.n27.2022.04
pISSN: 1390-650X / eISSN: 1390-860X

Evaluation of AIoT performance in Cloud and Edge computational models for mask detection

Evaluación de AIoT en modelos computacionales en la nube y en el borde aplicado a la detección de mascarillas

Felipe Quiñonez-Cuenca[1,*], Cristian Maza-Merchán[1], Nilvar Cuenca-Maldonado[1], Manuel Quiñones-Cuenca[1], Rommel Torres[1], Francisco Sandoval[1], Patricia Ludeña-González[1]

Received: 15-11-2021, Received after review: 20-12-2021, Accepted: 27-12-2021, Published: 01-01-2022

Abstract

COVID-19 has caused serious health damage, infecting millions of people and unfortunately causing the several deaths around the world. The vaccination programs of each government have reduced those rates. Nevertheless, new coronavirus mutations have emerged in different countries, which are highly contagious, causing concern with vaccination effectiveness. So far, wearing facemasks in public continues being the most effective protocol to avoid and prevent COVID-19 spread. In this context, there is a demand of automatic facemask detection services to remind people the importance of wearing them appropriately. In this work, a performance evaluation of an AIoT system to detect correct, inappropriate, and non- facemask wearing, based on two computational models: Cloud and Edge, was conducted. Having as objective to determine which model better suits a real environment (indoor and outdoor), based on: reliability of the detector algorithm, use of computational resources, and response time. Experimental results show that Edge-implementation got better performance in comparison to Cloud-implementation.

Keywords: AIoT, COVID-19, Cloud Computing, Edge Computing, Face mask detection, YOLO

Resumen

La COVID-19 ha provocado graves daños a la salud: centenas de millones de personas infectadas y varios millones de fallecidos en el mundo. Los programas de vacunación de cada Gobierno han influido en el decaimiento de estos índices, pero con la aparición de nuevas mutaciones del coronavirus más contagiosas, la preocupación sobre la efectividad de las vacunas se hace presente. Frente a esta situación el uso de mascarillas sigue siendo eficaz para prevenir la transmisión y contagio de la COVID-19. Lo que ha generado una creciente demanda de servicios de detección automática de mascarillas, que permita recordar a las personas la importancia del empleo de estas. En este trabajo se plantea un análisis del rendimiento de un sistema AIoT para la detección del uso correcto, incorrecto y sin mascarilla basado en dos modelos computacionales de Cloud y Edge, con la finalidad de determinar qué modelo se adecua mejor en un entorno real (interior y exterior) sobre la base de la confiabilidad del algoritmo, uso de recursos computacionales y tiempo de respuesta. Los resultados experimentales demuestran que el modelo computacional Edge presentó un mejor desempeño en comparación con el Cloud.

Palabras clave: AIoT, COVID-19, computación en la nube, computación de borde, detección de máscara facial, YOLO

[1,*]Departamento de Ciencias de la Computación y Electrónica, Universidad Técnica Particular de Loja, Loja, Ecuador. Corresponding author ✉: fdquinones@utpl.edu.ec.

Suggested citation: Quiñonez-Cuenca, F.; Maza-Merchán, C.; Cuenca-Maldonado, N.; Quiñones-Cuenca, M.; Torres, R.; Sandoval, F. and Ludeña-González, P. (2022). «Evaluation of AIoT performance in Cloud and Edge computational models for mask detection». Ingenius. N.° 27, (january-june). pp. 32-48. doi: https://doi.org/10.17163/ings.n27.2022.04.

1. Introduction

Humanity is experiencing a health crisis due to COVID-19, caused by the new SARS-CoV-2 coronavirus strain [1], which has unexpectedly and dramatically affected people's health, economy and lifestyle worldwide. The origin of the virus was identified at the end of 2019 in the city of Wuhan, China. Subsequently, the virus spread worldwide and on March 11, 2020 it was declared by the World Health Organization (WHO) as a global COVID-19 pandemic [2]. SARS-CoV-2 attacks people's immune systems. Although most people who get infected with the virus have mild and moderate symptoms, another significant group of people need hospitalization, and even the use of ICU beds (Intensive Care Unit). This virus is transmitted mainly through microscopic droplets of cellular plasma expelled by the infected person when coughing, sneezing, or exhaling [3].

Twenty months of pandemic have marked the history of human health, which is evidenced by official statistical data. To date, December 3, 2021, globally, according to WHO statistics (2021), a little more than two hundred and sixty-three million cases have been confirmed, of which more than five million died. On the continental level, America has the highest rate of infection in relation to the other five continents, nearly 97 million confirmed cases have been reported, of which more than two million people died. While in Ecuador, more than five hundred and twenty-seven thousand confirmed cases are recorded and more than thirty-three thousand have died [4].

To neutralize the COVID-19 pandemic, world leaders made crucial decisions, leading to collateral problems from which humanity is still recovering. Measures include global confinement, implementation of safety protocols, and even the invention and distribution of vaccines.

At the local level, in mid-March 2020, the Ecuadorian government declared a state of emergency, which involved restrictions on mobility, isolation, and border closures, which severely affected the country's economy. Companies had to abruptly halt their production, and government, educational and financial institutions were forced to continue their activities online.

After four and a half months of confinement, the country's economy did not resist and confinement restrictions were gradually eliminated; but biosafety protocols had to be implemented by the community to work face-to-face. To prevent the spread of SARS-CoV-2 virus, each individual is required: compulsory use of masks, avoid conglomerations, keep distance, wash hands regularly with soap, and continuously disinfect commonly used surfaces with alcohol. Strict monitoring of biosafety protocols has been the key to preventing the virus, since initially, there was no specialized medicine and vaccine to protect people because SARS-CoV2 was new. Several COVID-19 vaccines are currently available.

The Pan American Health Organization (PAHO) [5], on its official website, has made available to the general public information related to COVID-19 vaccines, including Pfizer/BioNTech, Moderna, AztraZeneca, Janseen, Sinopharm, and Sinovac. Although there are several vaccines, the vaccination process does not progress as planned. According to statistics [6], to date (December 2021) 54.9% of the world's population has received at least one dose of the COVID-19 vaccine. More than eight billion doses have been administered worldwide, and thirty-four million doses are administered each day.

Simultaneously with the global vaccination process, new variants of the COVID-19 virus have been identified in different regions of the world. The high number of infected people increases the risk of virus´s mutations. The more the virus spreads, the smaller changes occur in its genetic code, allowing it to survive and reproduce. Multiple variants circulate globally, for example: the UK variant, known as Alpha; the South African variant, known as Beta; the Brazilian variant, known as Gamma, the Indian variant, known as Delta; and the last one known as Omicron detected in South Africa. According to experts, viruses mutate all the time, and most changes are irrelevant, but others can make the disease more infectious or threatening, and these mutations tend to dominate [7].

The goal of vaccination is to achieve global collective immunity to prevent SARS-CoV-2 from continuing to mutate, becoming more resistant to current vaccines and causing more periods of mass mortality. However, PAHO, at the end of the first week of August, due to a high rate of contagion of the variant (VOC) Delta, in several countries, inside and outside America, recommends reviewing contingency plans and be prepare for an eventual increase in cases and hospitalizations, emphasizing in that report that social distancing, the use of masks, and the use of antiseptic solutions continue to be the most effective measures to reduce transmission of this and all variants. From the above data, it can be deduced that COVID-19 pandemic has not yet ended, the future is uncertain. Research and innovation are therefore needed to provide technological contributions to society in the fight against COVID-19 pandemic.

This paper describes the design, implementation, and performance analysis of a system based on two cloud and edge computing models, applied to real-time mask usage detection by using artificial intelligence of things (AIoT). The importance of system development and evaluation lies in the fact that it analyzes response time, detection algorithm performance, and computational resources.

The article is organized as follows. Section II describes the proposed method, and details the computational designs and components used. Section III

presents the results of the analysis of computer models. Finally, Section IV shows the conclusions and future work.

2. Materials and methods

The research was divided into five phases (see Figure 1) through an experimental design and quantitative approach. Then, the activities that were carried out in each of the phases are detailed:

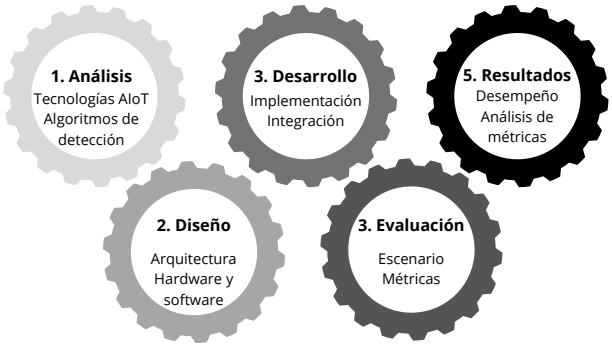


Figure 1. Methodology

- State-of-the-art analysis of enabling technologies for AIoT, related works and detection algorithms to detect the use of masks in real time.
- Design of an architecture and determination of hardware and software components for deployment of computing models in the cloud and on the edge of the network.
- Development in the stage in which software and hardware components are integrated, and the real-time mask detection and execution algorithm implemented in both computational models.
- Evaluation of the stage in which a controlled scenario and real scenarios are determined to perform tests that allow validating the performance of the computational models.
- Analysis of results to determine the performance of resources used in both computer models. Metrics collected include the demand for computing resources (CPU, RAM, memory, and storage), and system response time. To analyze the performance of the detection algorithm, the accuracy, precision, revocation, harmonic mean, and mean of the average accuracy are evaluated.

2.1. Architecture

For developing the system, a comparison strategy was designed to allow the two computer models to be equipped with the same or similar capabilities. However, because each implementation has intrinsic characteristics that differentiate them from each other, the comparison strategy was designed with two structures, one common to both computational models and the one specific to each one. In Figure 2 can be observed the layout of each of the hardware and software components that are part of the system architecture, segmented into: scenario, sensors and actuators, computational models, and actuators.

The system architecture integrates an IP camera to obtain the video from the test scenario, which is sent via Real Time Transmission Protocol (RTSP) to the two computer models. Video processing is performed by applying the real-time object detection algorithm based on YOLOv3 («You only look once v3») to determine correct and incorrect use and no mask. The Edge computational model performs real-time inference and presents the results on a screen by ticking the faces depending on the result with «correct mask», «incorrect mask» and «no mask». Audible and visual alarms are generated, and it is then sent to a web platform in the cloud to store the processed images. While the processing is carried out in the Cloud computational model and the result can be visualized through a device and the inference is stored as detailed in the Edge model.

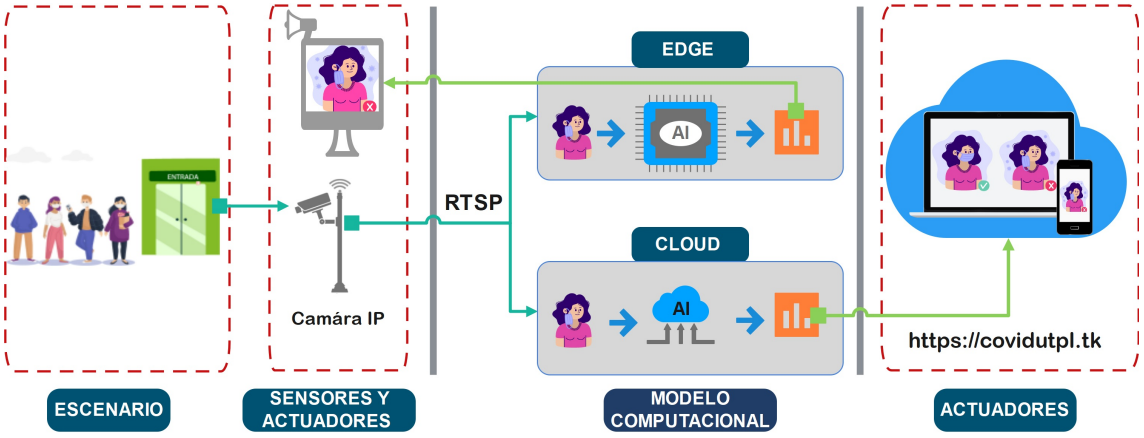


Figure 2. System architecture

2.1.1. Common structure for Cloud and Edge

According to [8], comparative analysis is emphasized in the broad explanation of similarities and differences of phenomena to provide valid reasons in a topic or area of interest. Consequently, this section describes the "similarities" of the comparison strategy between Cloud and Edge computing models. To evaluate the performance of the AIoT algorithm, it was discussed the use of the same transmission protocol, codec video, visual sensor (IP camera), training dataset, training algorithm, evaluation dataset, and execution algorithm for real-time mask detection in both models

2.1.2. Transmission protocol

Currently, there are several streaming protocols such as Real Time Streaming Protocol (RTSP), Real-Time Messaging Protocol (RTMP), Secure Reliable Transport (SRT), and WebRTC (RTC – Real Time Communication) [9]. In selecting the transmission protocol to be used in the comparative strategy of this research, the results of recent research and the compatibility of the transmission protocol with the surveillance video cameras available in the market were taken into account. According to [10], one of the most widely used streaming video protocols is RTSP, especially in environments with bandwidth restrictions, network congestion, energy efficiency, cost, reliability, and connectivity. Whereas, according to [11], the RTMP protocol offers better performance in live video streaming compared to the RTSP protocol.

When reviewing the compatibility of transmission protocols with the visual sensors (video surveillance cameras) offered in the market, cameras were found with two RTMP and RTSP protocols. To implement the RTMP protocol push mechanism, another component (a streaming server) must be added to the network, altering the flow of data across the network and the comparison architecture, and impacting the traffic analysis result. On the other hand, the RTSP protocol pull mechanism is a convenient option to implement since both Cloud and Edge computing models can be connected to the same visual sensor (IP surveillance camera). Therefore, RTSP was selected as the transmission protocol for the comparative strategy, because it is better suited to the proposed comparison architecture.

2.1.3. Codec video

According to [11], the H.264 codec video is one of the most functional compression standards in Internet of Things (IoT) applications because it occupies less capacity when stored or transmitted. Also, the video compression standards of the H.264 codec are based on motion compensation. This codec is highly recommended for recording, compressing and distributing video files in real time [12]. Also, the H.264 codec video supports the RTSP protocol and is available in many of the visual sensors offered on the local market. Finally, based on the latter, it is determined that the H.264 codec is compatible with the proposed architecture.

2.1.4. Mask detection algorithm

AIoT combines two approaches, namely IoT and Artificial Intelligence (AI). [13]. IoT approach refers to the concept of interconnecting objects to the network, so that information can be collected through sensors automatically, without the exclusive intervention of people [14]. While AI refers to the approach of providing intelligence machines through algorithms, so that they can make decisions based on previously received training [15]. By uniting both approaches, an attempt is made to adhere a cognitive layer to the network, to achieve resource optimization through the autonomy that can be provided to machines to analyze situations and make decisions.

Media processing is a major challenge for AIoT

algorithms due to the large amount of data that this activity involves and due to the limited resources of IoT. In this case study "real-time mask use detection", the aim is to evaluate the performance of AIoT algorithms in Cloud and Edge computing models. For which, in the state-of- the-art it was identified that the technical solution for this type of object recognition problems is AIoT algorithms, based on Deep Learning with CNN (Convolutional Neural Network) architectures [16].

In terms of real-time object detection, the literature highlights the detector models of a stage for better performance compared to two stages. YOLO algorithm, a single-stage detector model, is characterized by a significant difference in speed compared to the two-stage R-CNN (Region Based Convolutional Neural Networks) and Fast-R-CNN models. YOLO is a thousand times faster than R-CNN and a hundred times faster than Fast-R-CNN [16]. In contrast to the approach adopted by YOLO's predecessor object detection algorithms, which reuse classifiers to perform detection, YOLO proposes the use of an end-to-end neural network that makes predictions of bounding boxes and class probabilities simultaneously in a single iteration [14].

For the present strategy of comparing the performance of AIoT algorithms between the Cloud and Edge computational models, it was decided to select the reduced version of YOLOv3, identified as YOLOv3-tiny for two reasons. The first is that this version is a small model ideal for deployments where computing resources are limited; i.e., it is compatible with the Edge deployment architecture, which is conducive to equitable comparison. While the second reason is because it is the latest version recognized as official, which means that it has access to the official support of the development community. YOLO model is widely implemented in solutions that require real-time object identification, due to its architecture and operation [16–19].

YOLOv3 was proposed as a solution that uses modern CNN, which uses residual networks and omits connections. According to [14], authors of YOLOv3, this version uses the much more complex Darknet-53 convolutional neural network as the model's spine, 106-layer with residual blocks and superior sampling networks. This architecture enables the YOLOv3 model to predict at 3 different scales, and extract feature maps at layers 82, 94, and 106 for these predictions.

2.1.5. Dataset

A training and validation image set is required for the detection algorithm preparation process. This work requires training the real-time object detector algorithm YOLOv3-tiny, so that it learns and detects three classes. In other words, for the implementation of the present case study "detection of mask use" the detection algorithm of people with 1) correct use of mask, 2) no mask and 3) incorrect use of mask will be trained. Figure 3 illustrates the dataset with the three required classes.

Figure 3. Dataset with three types: correct use of mask, no mask, incorrect use of mask

The dataset must contain information relevant to the context, i.e., images of people using masks of different colors and models for the first class; images of people of different ages and ethnicity for the second class; and images of people using the mask incorrectly, below the nose or mouth, for the third class.

For algorithm training, the dataset was customized by randomly selecting images from two public datasets Kaggle Medical Mask Dataset [20] and Masked Face (MAFA) dataset [21], because such datasets contain real images of people in different backgrounds, unlike other datasets that contain all three classes, but correspond only to the person's face. According to [17], in-context information is another approach used to improve detection accuracy or speed. Additionally, for the training of the detector algorithm, it is planned to divide the custom dataset according to the hold-out technique, which consists of dividing the dataset into two subsets of 80% for training and the remaining 20% of the data for testing [22]. In this way, the performance of the algorithm on unseen data can be evaluated after training [23]. While, in the execution phase of the training of the neuroneal network specialized in the detection of masks, the technique «transfer learning» is used, which consists of transferring knowledge of a pre-trained model in a general context to a more specific one [24], i.e., for the detection of the 3 classes (with mask, without mask and incorrect mask) the base model of pre-trained YOLOv3-tiny will be used for detecting 80 classes of objects.

2.2. Execution algorithm in real time

Algorithm 1 describes by pseudo code the process for detecting all three classes (correct use of mask, no mask, and incorrect use of mask). This algorithm has

as input the flow of images (video stream) captured by the visual sensor of the network, while the output will be the detection(s) of people classified in the three options.

Before the detection process, it is necessary to prepare each frame of the video stream, by resizing the frame to multiple scales based on the YOLOv3-tiny architecture. Each frame is converted to RGB (red, green, blue) channels. Finally, the frame and its capture date are stored. All detections are then stored in the "Face" array using YOLOv3- tiny. Then, it is necessary to evaluate if there is any detection in the current frame; i.e., if the "Face" array contains any elements (higher than zero).

In case that there is no detection in the current frame, the process is completed, and the next frame of the video stream is worked on. Whereas, if there is at least one detection in the face array, it is required to evaluate for each detection the class to which it belongs to, to highlight its bounding box and assign the corresponding class label (ID - identification). In case that: 1) If the ID detection belongs to the "with mask" class, the green bounding box is highlighted; 2) if the ID detection belongs to the "without mask" class, the red bounding box is highlighted; and 3) if the ID detection corresponds to the "wrong mask" class, the orange bounding box is highlighted. In addition, for each detection, the detection bounding box must be cut and stored in JPG format with the date and time of detection.

Algorithm 1 Mask detector from real-time video.

RTSP stream. Detection of mask **foreach** *<Frame in Stream>* **do**
- Redimension *Frame*;
- Convert *Frame* to RGB;
- Store date and time;
- Identify *Faces* in the *Frame* using YOLOv3-tiny;
- **if** *Faces* > 0 **then**
  - **foreach** *Face in Faces* **do**
    - **switch** *Face* **do**
      - **case** *Correct use of mask* **do**
        - Highlight the *Frame* in gree
      - **case** *No mask* **do**
        - Highlight the *Frame* with red
      - **case** *Incorrect use of mask* **do**
        - Highlight the *Frame* with ORANGE
    - Cut *Face* and store *Frame* as a JPG file;
    - Store *Face*, *Dateandhour* in the database;

2.3. Edge architecture design

Figure 4 illustrates the architecture design of the Edge computational model. The central processing unit is the Jetson Nano component, which has as input the video stream captured by the network's visual sensor (TPLINK tapo C310 V2 IP camera).

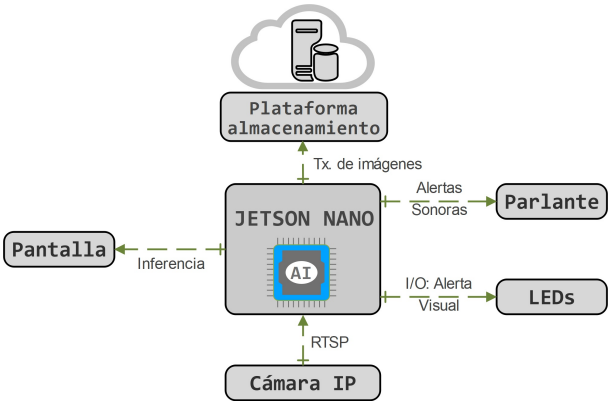


Figure 4. Architecture of the Edge computational model

The AIoT algorithm for real-time mask usage detection is executed on the central processing unit. Subsequently, four outputs occur on the system actuators in case of a detection: 1) the transmission of the images corresponding to the detections to a cloud repository, 2) an audible alert that highlights the detection class, 3) the activation of LEDs depending on the detected class, and 4) the display of the detection on a screen. Finally, Figure 5 shows the inference results in the Edge computational model.

Figure 5. Inference results in the Edge computational model

2.4. Cloud architecture design

Cloud platforms allow end users to deploy their AIoT solutions based on real-time video streaming, analysis, and storage, including Amazon Web Service (AWS) Kinesis Video Stream (KVS) and SafetyRadar [25]. The first platforms are characterized by offering different types of object detection, for example, houses, cars, animals, etc. The detection of different objects to be implemented requires technical knowledge to use these platforms, while the SafetyRadar platform specializes in the detection of masks and other biosafety implements, offering plug and play technology, hence

no technical expertise is required for deployment. However, all three platforms are paid.

For the present work, a custom architecture was designed for the cloud computing model, with control over the different network components. Generally, platforms do not allow the extraction of metrics required for comparison, which is the aim of this investigation.

Figure 6 illustrates the architecture design of the Cloud computational model. The video stream captured by the network's visual sensor is accessed via the RTSP transmission protocol from the local network. While two components are arranged in the cloud using a container architecture, specifically Docker [25], enabling the deployment and replication process of the deployment to be streamlined. The first container processes the video stream, while the second container publishes the results of the processing. In container 1, after the multimedia data, the input data is processed according to business logic, with the execution of the AIoT algorithm for the detection of mask use in real time. The results of such processing are then stored; in this case, the images with the detections performed. Finally, events made in a log are saved. Meanwhile container 2 presents the results so that end users can connect to the server and use the information they want through a Website.

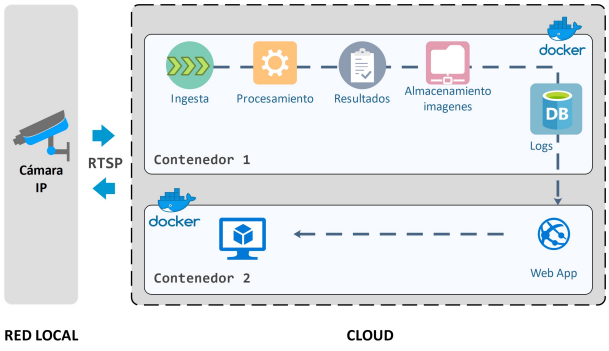


Figure 6. Architecture of Cloud computational model

Figure 7 shows the inference results of the Cloud computational model.

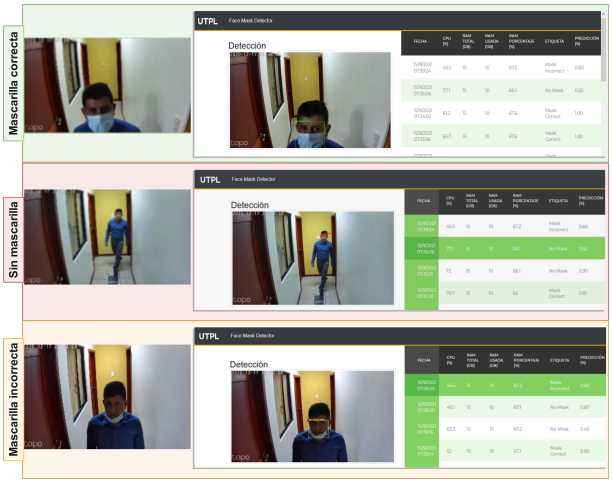


Figure 7. Inference results of the Cloud computational model

2.5. Technology used

Table 1 shows a summary of the technology used in this research. The first column identifies the technology used, the second column corresponds to the Cloud or Edge computational model where the technology is applied; the third column indicates which component to use. The last column provides justification for the use of each technology.

Table 1. Summary of technologies used

Tecnology	Model	Component	Justificación
JetPack SDK 4.5.1 [26]	Edge	Processing	Includes Jetson Linux Driver Pack (L4T) with Linux operating system and CUDA-accelerated libraries (TensorRT and cuDNN), for AI APIs.
Python3 [27]	Edge Cloud	Processing	Base language for video processing, results publication, and integration with vision and AI libraries.
Deepstream SDK 5.1 [28]	Edge	Processing	This Nvidia SDK provides a framework for building GPU-accelerated video analytics applications that run on the NVIDIA Jetson Nano platform.
Deepstream-services-library [29]	Edge	Processing	Python3 library required to make use of the deepstream SDK functionalities.
MySQLdb [30]	Edge	Results	Provides the Python API to access the MySQL database server.
rclone [31]	Edge	Results	Required to manage files in cloud storage.
Docker [32]	Cloud	Processing	It allows automating the deployment of the application in the containers located in the cloud.
Camgear [33]	Cloud	Processing	It is a high-performance, cross-platform video processing framework that enables real-time video processing.
OpenCV [34]	Cloud	Processing	Library that allows to process artificial vision in Python3. This facilitates in the first instance the reading of the trained model of YOLOv3-tiny, and later in the detection for each processed frame.
Firebase [35]	Cloud	Processing	It allows to store the Results of the detections in real time.
Flask [36]	Cloud	Results	Framework that is used for the creation of the web application.
Bootstrap [37]	Cloud	Results	Library for the design of the application that allows publishing the Results.

2.6. Experimental configuration

The image tagging and custom dataset splitting processes to be used in mask detection algorithm are

performed using the Python3 programming language. The dataset was divided with a distribution of 80% (716 images) for training and the remaining 20% (179 images) for testing (see Figure 8). The images in the custom training dataset contain the detection of one person or more people in the training, representing 3070 people with correct use of mask, 675 without mask and 113 with incorrect use of mask.

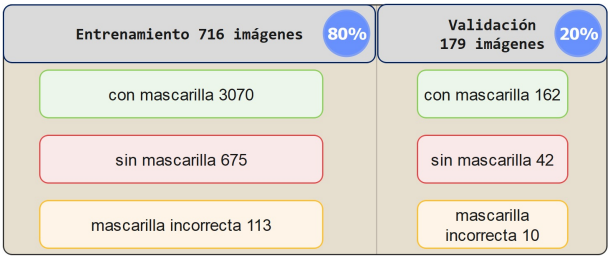


Figure 8. Dataset distribution for training and validation

The Google Colab [38] tool was used to run the AIoT algorithm training process, which has three inputs: 80% of the dataset images, YOLOv3-tiny configuration file (see Table 2 for configuration parameters), and YOLOv3-tiny pre-trained weight file with COCO dataset [39].

Table 2. Configuration training values for neuronal network

Detail	Calculus	Value
Number of classes	Correct use of mask, no mask and incorrect use of mask	classes=3
Maximum batch size	30000 (10000 per class)	max_batches = 30000
Dimensions of input image	width × height = 416 × 416	width=416 height=416
Number of filters in the convolutional later before YOLOv3-tin layer	(classes+5)*3 = 24	filters=24
Batch size and its subdivisions for training		batch=64 subdivisions=2

The training process lasted 16 hours, obtaining a mean accuracy (mAP) of 75.95%. Finally, the trained neuronal network was validated with the remaining 20% of the dataset (79 images), representing 162 people with correct use of mask, 42 without mask and 10 with incorrect use of mask. Table 3 shows the validation results.

Table 3. Validation results of the detection algorithm

Class	Precisión
Correct use of mask	85,97 %
No mask	68,72 %
Incorrect use of mask	73,15 %
mAP	**73,15 %**

3. Results and discussion

This section presents the results of the different experiments in two real scenarios to determine the performance of the two computational models. In this context, two environments were selected. The first indoor scenario corresponds to the entrance of people to a local church to participate in the Sunday Eucharist. For the second outdoor scenario, it was applied on a street with a high rate of pedestrian circulation, due to religious tourism to the Sanctuary of El Cisne between August and September, located in the province of Loja, southern Ecuador.

In both indoor and outdoor environments, it was monitored for a period of one hour, of which for a more detailed analysis the first 15 minutes are taken for the quantitative analysis of the results. In this context, 82 people entered indoors and 229 circulated outdoors. Figure 9 illustrates the total number of detections in indoor and outdoor environments.

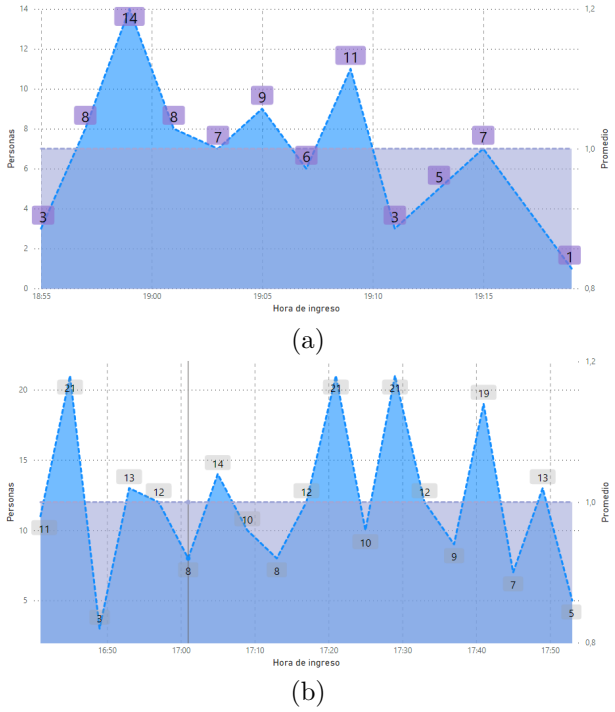


Figure 9. Flow of people: a) indoor, b) outdoor

The results of the experiment show that the flow of people in the outdoor environment doubled the indoor, having a mean income of 6 people in the internal environment and 12 people in the external environment. The count was carried out through the Camlytics tool. Figure 10 shows detection examples in the real indoor and outdoor scenarios.

3.1. Evaluation metrics

To evaluate the performance of the AIoT algorithm in Cloud and Edge computing models, the "Real-time Biosafety Mask Usage Detection" case study has been considered as evaluation metrics:

- Detection algorithm reliability: According to [16], [18], [40–42], determining the performance of the detection algorithm is based on four metrics: accuracy, precision, sensitivity, harmonic mean, mean of the average accuracy.
- Computational resources: it is recommended to analyze the resources occupied by the application based on [41]] and the ISO/IEC 25023 standard, analyzing network traffic, RAM memory, CPU and storage.
- Response time: This is another parameter that allows to measure the time it takes for the system to respond according to [41] and by ISO/IEC 25023 standard.

Figure 10. Indoor and outdoor live stage results

3.1.1. Accuracy of detection algorithm

To evaluate the accuracy of ML-based classifiers (Machine Learning) a confusion matrix is used to know how effective the system is. According to [40], a confusion matrix, also known as an error matrix, is a method for 10 to summarize the performance of the result of a classification model, showing the number of correct and incorrect predictions. Four metrics are calculated from the confusion matrix [16], [18], [40–42]: Accuracy (A: Accuracy, see Equation 1), precision (P: precision, see Equation 2), sensitivity or recall (R: Recall, see Equation 3), and the harmonic mean (f_β, see Equation 4). Additionally, three classes are identified due to the AIoT Edge and Cloud implementations of this experiment (MC: Correct use of mask, SM: No mask and MI: Incorrect use of mask), it is also necessary to calculate the average of the mean accuracy (mAP, see Equation 5) from the mean accuracy (AP) of each class.

$$A = \frac{TP+TN}{(TP+FP)+(TN+FN)} \quad (1)$$

$$P = \frac{TP}{TP+FP} \quad (2)$$

$$R = \frac{TP}{TP+FN} \quad (3)$$

$$f_\beta = \frac{(1+\beta^2)*(P*R)}{\beta^2*P+R} \quad (4)$$

$$mAP = \frac{AP_{MC}+AP_{SM}+AP_{MI}}{3} \quad (5)$$

Where: TP (True Positive) is when the observation is positive and is predicted to be positive; FN (False Negative) is when the observation is positive, but is predicted to be negative; TN (True Negative) is when the observation is negative and is predicted to be negative; and FP (False Positive) is when observation is negative, but is predicted to be positive, while β is used to assign different weights to the measures used in Equation 4 [42]. In the present work, β was assigned a value of 1. Table 4 shows the results obtained when deploying Edge and Cloud models in real-world scenarios. This is done by purging repeated detections in each environment.

Table 4. Confusion matrix

Edge model indoors				
		True value		
		MI	SM	MC
Prediction	MC	2	29	**191**
	SM	0	**36**	9
	MI	**3**	3	0
Edge model outdoors				
Prediction	MC	2	15	**170**
	SM	0	**52**	30
	MI	**1**	1	0
Cloud model indoors				
Prediction	MC	5	36	**165**
	SM	7	**29**	12
	MI	**1**	2	6
Cloud model outdoors				
Prediction	MC	3	45	**123**
	SM	2	**41**	40
	MI	**1**	2	2

Table 5 shows the results obtained after evaluating the Edge and Cloud implementations for detecting the use of masks by individuals in both indoor and outdoor environments, in time and real-world scenarios.

Table 5. Results obtained for A, P, R and f_β

Accuracy (A) %				
Clase	**Cloud**		**Edge**	
	indoor	*outdoor*	*indoor*	*outdoor*
MC	76,7	64,7	85,2	82,6
SM	77,4	65	84,9	82,9
MI	90,7	94,8	97,9	98,7
Media	81,6	74,8	89,3	88,1
Precision (P) %				
MC	80	71,9	86	90
SM	60,4	49,4	80	63,4
MI	11,1	20	50	50
mAP	50,5	47,1	72	67,8
Recall (R) %				
MC	90,1	74,5	95,5	85
SM	43,3	46,6	52,9	76,5
MI	7,6	16,7	60	33,3
Media	47	45,9	69,5	64,8
Harmonic mean (f_β) %				
MC	84,8	73,2	90,5	87,4
SM	50,4	48	63,7	67,9
MI	9	18,2	54,5	40
Media	48,1	46,5	69,6	65,1

Table 4 shows the percentages obtained with respect to the accuracy, precision, recall, and harmonic mean metrics. Accuracy refers to the number of positive predictions that were correct; in this context, the results show that the Edge model is 10.5% more accurate than the Cloud model. On the other hand, precision refers to the percentage of positive cases detected; the results indicate that indoor accuracy is higher than outdoor in both models. The Edge is 39.9% more accurate than Cloud. Cloud accuracy is lower than Edge as some frames are lost or distorted during transmission in some cases. Regarding the recall metric, which refers to the proportion of positive cases that are correctly identified by the algorithm, the results indicate that the Edge model predicted 20.7% more correct than Cloud. Finally, the harmonic mean metric was used when the dataset is not balanced by providing inputs from different classes to the classifier. In this context, the results show that insufficient detections of the "incorrect use of mask" class have a dramatic impact on data distribution.

3.2. Use of resources

Among the resources analyzed for comparing the two Cloud and Edge computing models according to [41] and ISO/IEC 25023, are network traffic, RAM, CPU, and disk storage usage. Since the results indicated a better development of the masks detector indoors, an analysis of the use of resources in this environment is performed in the following sections. In addition, Table 6 shows the characteristics of the hardware components on each of the two models, Cloud and Edge.

Table 6. Hardware characteristics of Cloud and Edge

Component	**Cloud**	**Edge**
GPU	—	NVIDIA Maxwell architecture with 128 NVIDIA CUDA® cores
CPU	Intel Xeon Processor (Skylake, IBRS) / 8 núcleos / 2100 MHz	Quad-core ARM Cortex-A57 MPCore processor
Memory RAM	31 GB virtual	4 GB 64-bit LPDDR4, 1600MHz 25.6 GB/s
Storage	246 GB	16 GB eMMC 5.1
Connectivity	Gigabit Ethernet	Gigabit Ethernet, Wi-Fi
Screen	—	HDMI 2.0 and eDP 1.4

3.2.1. Network traffic

Network traffic measurements generated by streaming video from the IP camera on the Cloud model, and images from the Jetson Nano on the Edge model were taken using the Wireshark protocol analyzer on the inbound interface of the server hosting the Cloud model and the Web platform. Traffic measurements were obtained using video encoding parameters at 1008 Kbps, observing the behavior of the traffic in Mbps volume. It can be observed in Figure 11 the network traffic generated, where the blue upper curve represents the traffic from the IP camera to the Cloud computing model using the RSTP protocol; the lower green curve represents the traffic generated by the transmission of images processed with the detection algorithm on the Edge to the web platform (measured in Mbps). This information is represented as series over time, which corresponds to the first 15 minutes of indoors monitoring.

As a result, it can be seen that network traffic is significantly higher in the Cloud model compared to Edge model. The average network traffic in the cloud is 0.86 Mbps, with a maximum of 4.91 Mbps; while the average network traffic in the Edge is 0.07 Mbps, with a maximum of 0.26 Mbps. This is because all frames of the visual sensor in the Cloud model are transmitted without pre-filtering information; in the Edge model, images are only transmitted when there is a detection, which prevents network congestion.

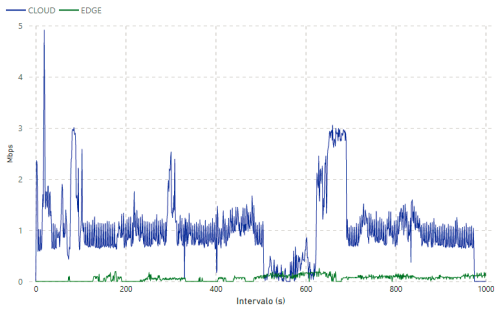


Figure 11. Video transference rate to the Cloud and processed images from the Edge to the Web platform

3.2.2. Memoria RAM

Los modelos computacionales Cloud y Edge se caracterizan por la diferencia en la cantidad de recursos disponibles. Particularmente, en las implementaciones de este trabajo se dispuso de 4 GB de RAM en el modelo Edge y de 31 GB de RAM en el modelo Cloud. Los resultados indican (ver Figura 12) que el uso de RAM en el modelo Edge fluctuó entre 2,41 GB como mínimo, 2,43 GB como máximo, con una media de 2,42 GB. Mientras que en el modelo Cloud, el uso de RAM fluctuó entre 10,7 GB como mínimo, 12 GB como máximo, con una media de 11,55 GB. A partir de los datos obtenidos, se puede interpretar que, una vez designados los recursos al proceso YOLO, no existe mayor crecimiento durante la ejecución de las detecciones tanto en Cloud como Edge.

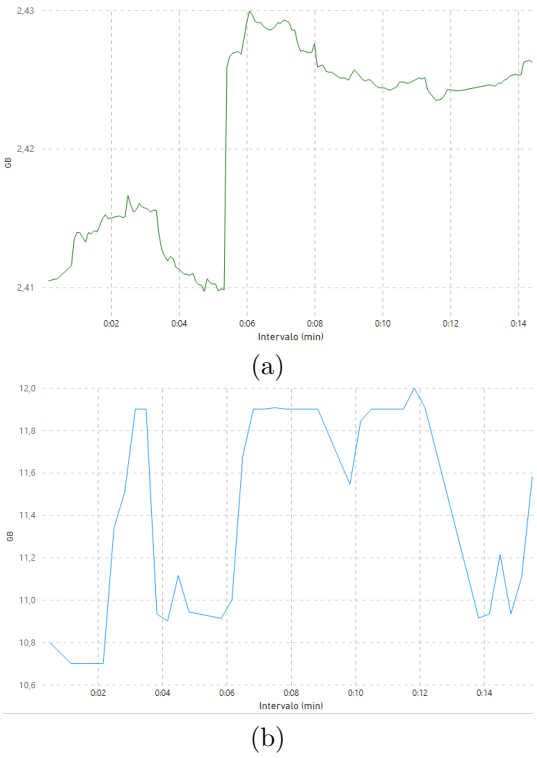


Figure 12. Use RAM memory during the first 15-monitoring minutes indoors. A) Edge model, b) Cloud model

3.2.3. CPU

Regarding the processing capacity of the Cloud and Edge computing models, they differ significantly. From Figure 13, it is determined that the Edge model with the Jetson Nano integrates a 1.43 GHz processor, of which CPU use during the first 15 minutes of processing fluctuated between 22.95% minimum, 62.01% maximum, and an average of 29.59%. Whereas in the Cloud model, the server had a 4.8GHz processor, of which CPU utilization fluctuated between 1.75% minimum, 7.49% maximum, and an average of 3.31%. Therefore, it can be interpreted that because the Edge model has fewer resources, the effort is greater at processing detections. Conversely, because the Cloud model has better resources, the effort is minimal when running the mask usage detector.

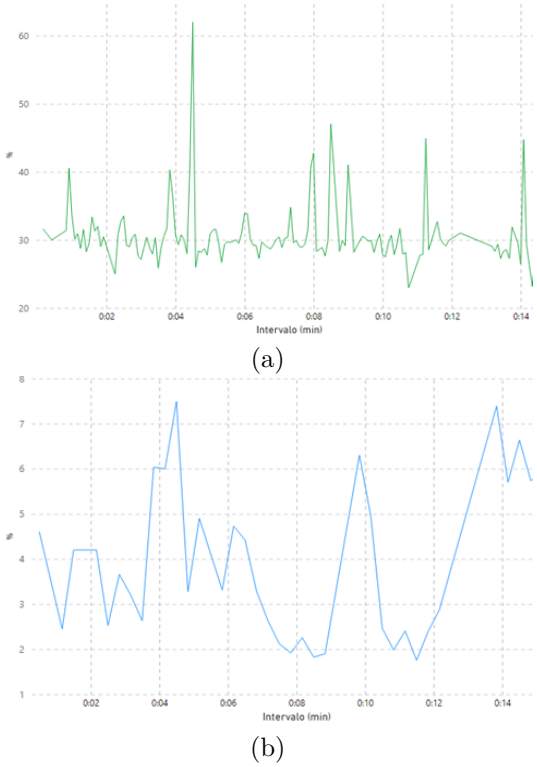
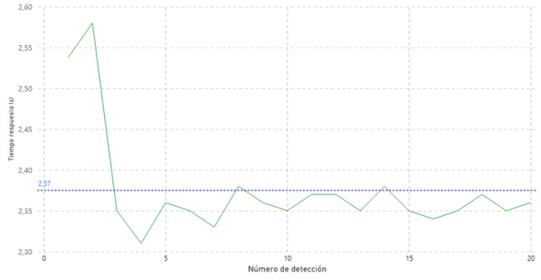


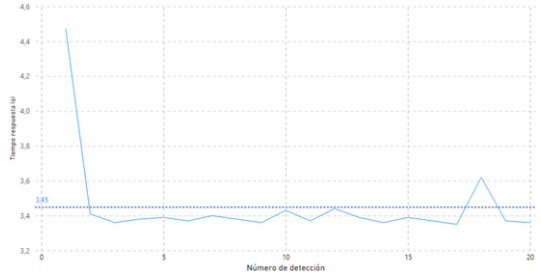
Figure 13. CPU use during the first 15-monitoring minutes indoors, a) Edge model, b) Cloud model

3.2.4. Storing

Disk space usage of detections grows significantly in both Cloud and Edge models. In the Cloud, this resource can increase in case of a larger demand by updating the cloud server lease. Conversely, Edge does not allow any increased storage capacity as it is limited in nature. The results show that disk space usage is similar in both Cloud and Edge models, approximately 90.7 MB indoors and 200.5 MB outdoors. It is important to note that the mask detector stores only the detection frame not the context or the entire image, so the storage capacity is optimized.



(a)



(b)

Figure 14. Response time, a) Edge model, b) Cloud model

3.3. Response time

According to [41], and the ISO/IEC 25023 standard, the response time of a system is another metric that allows to evaluate the performance of a system. In this research, the response time value was used to obtain the events marked in the log files to identify the instant the image entered the model, and then to calculate the time elapsed until its publication. In this context, to calculate the response time in the Edge implementation, the time between the moment the detection occurs until the system finishes with the sound alert was considered. Figure 14a graphically shows the response times recorded on the Edge during the first 20 detections indoors, averaging 2.37 seconds of response.

On the other hand, measures from the time the capture occurs on the visual sensor to the publication of the results on the web page must be done to calculate the response time in the Cloud model. However, what it is posible to capture accurately is the time span from cloud discovery to portal publishing; so, in the Cloud time an average value of 2 seconds was added from the transfer rate for a 300 kB image with a 1920 × 1080 resolution, which corresponds to the time it takes for the visual sensor to transmit the video to the cloud, which is indirectly proportional to the bandwidth provided by the network. The bandwidth that was available for the implementation of the Cloud model was 25Mbps. Figure 14b graphically shows the response times recorded in the Cloud model during the first 20 detections indoors, averaging 3.45 seconds of response.

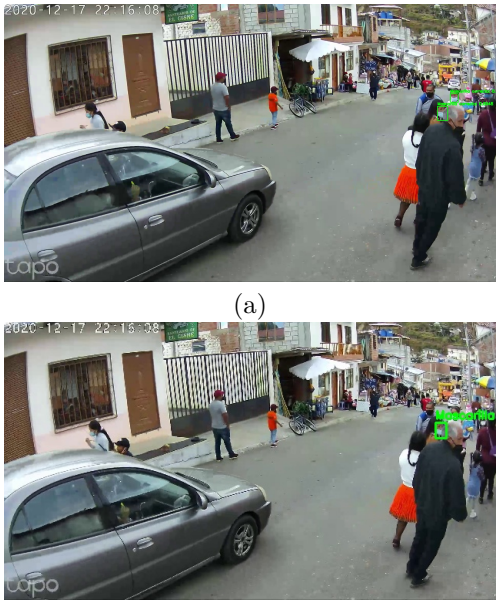
3.4. Discussion of the results

After deploying the Cloud and Edge implementations in real-world scenarios, it is observed that the AIoT algorithm (specialized in identifying the use of biosafety masks) performed better in indoors on both models due to controlled flow (in one direction) and exclusive of people who constituted the entrance to the church. While the performance of the same AIoT algorithm was less efficient in outdoors in both implementations, due to the uncontrolled flow of people (in several directions) and not exclusive of people; other objects such as cars, bicycles, and even animals were part of the images captured by the visual sensor. AIoT algorithms for real-time object identification, using the Edge computational model, presented better performance compared to the Cloud model.

YOLO is a recommended tool for implementing solutions using AI in real-life scenarios that require immediate action. In particular, the results of this work yield 78.2% accuracy metrics on the Cloud model and 88.7% on the Edge model. Although there is a decrease in the classifier accuracy with 48.8% in the Cloud and 69.9% in the Edge, it was proven that the accuracy in both models reduces especially outdoors, due to the fact that the neural network was not sufficiently trained to identify small objects. Additionally it was noted that the detector also reduces when there are multiple people in the same frame to be detected in both models. To solve this problem a greater number of images containing multiple detections could be included in the training dataset.

The quality of the media flow and input for video processing in Cloud and Edge deployments impact

the performance of the AIoT algorithm. The real-time mask detector was tested with two visual sensor resolutions, high (1920 × 1080) and low (640 × 360), allowing to conclude that detection fails on small objects at a low resolution; however, large (nearby) objects are correctly identified in real time (see Figure 15), since frame processing does so smoothly.



(a)

(b)

Figure 15. Mask detection at different resolutions: a) 1920 × 080 y b) 640 × 360

The relationship between video streaming speed (frames per second) and its relevance in the business context of the AIoT solution needs to be evaluated. In this research, it was observed that it was not necessary to evaluate all the frames of the input. The AIoT algorithm first processed all received frames (15 frames per second), and too many detections belonging to the same person were obtained as output. Hence, it was concluded that it was necessary to filter the number of frames input to the detector (1 frame per second) and thus avoid processor saturation. In other words, the resources required to process each of the frames are unnecessary. To overcome this problem, frames per second are reduced, which consists of processing a certain frame for a certain interval. This resulted in optimization of processing resources when filtering input and storage output, as the number of detections for the same person dropped significantly, for example, from 1 GB to 100 MB.

Finally, having experimented with both Cloud and Edge models, several challenges are identified for future work. There is inconsistency between the detection speed of Machine Learning based sorting algorithms (ML) and the technology that allow the storage and management of actuators on the network. For example, the biosafety mask usage detector works at a rate of 15 detections per second, but when wanting to store this information, the transaction takes one detection per second, thus causing congestion. Current database managers do not reach the speed required when working with AI-based classifiers. The same happens with visual or auditory actuators; it does not matter AIoT algorithm to detect multiple objects simultaneously if the time it takes for the speaker or display to show that information to the public requires one second per object.

3.5. System evaluation in a real environment

To assess the effectiveness and feasibility of the proposed method, a prototype system equipped with YOLOv3 Tiny is presented, which can be deployed at the entrances of public places. The prototype system based on the Edge computational model is illustrated in Figure 16, as well as the integration and implementation of the various hardware and software components, including an IP camera, a Jetson Nano computer, and a display with HDMI interface

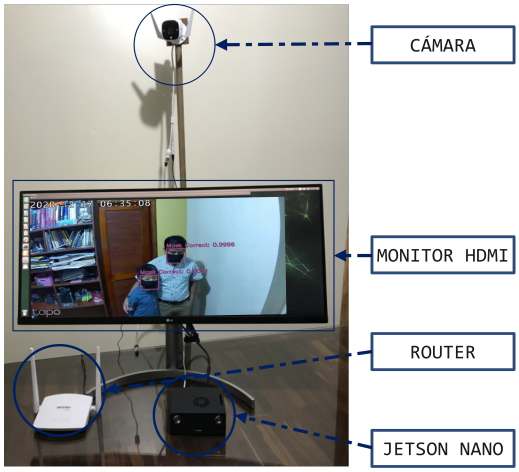


Figure 16. Implementation of the system

In Figure 5, the performance of the prototype can be proved for all three cases (correct use of mask, no mask, and incorrect use of mask). Also, visual alerts in the evaluation scenarios did not get people's attention when the AIoT algorithm provides detection feedback. Initially when people crossed the camera's viewing angle, the screen displayed the output of the labeled and painted bounding boxes with the class detected. It was evident that the purpose of the screen was not clear to the passerby, so some would greet or leave the camera's action area. This changed when adding the sound alerts that got people's attention. Users heard the following messages: "correct use of mask ", "no mask detected", or "incorrect use of mask"; depending on the type of detection performed by the algorithm, pedestrians identified the purpose of the deployment

intuitively, and many of them corrected the incorrect use or absence of the mask.

4. Conclusions

In this research, a system for detecting masks in real contexts has been designed, implemented and evaluated. The Edge computational model outperformed the Cloud model with 39.9% in the accuracy metric, related with the percentage of positive cases detected; and 10.5% on the accuracy metric, which refers to the percentage of positive predictions that were correct. In addition, there are advantages and disadvantages inherent to the intrinsic characteristics of each model. One of the advantages of cloud deployments is that they have high storage and processing capabilities, and if needing to increase RAM, disk, or processing resources, the model offers the scalability feature to the system. However, the disadvantage of the Cloud model is the high consumption of the network, since the sequence of frames is transmitted in video without prior data filtering. While in the Edge model, despite having limited resources, this disadvantage is offset by the fact that it is possible to perform a debugging of information, allowing applications to adjust to the amount of available RAM, processing, and bandwidth resources, where it could be observed that once the resources are allocated, the consumption of these resources does not fluctuate drastically.

In addition, on this side of the network, visual or auditory actuators can be used for human interactions in the context in which the model is being used. YOLOv3 application is an appropriate option for specialized real-time object detection in either the Cloud or Edge computational model. YOLOv3, as a single-stage object detector model, directly classifies and predicts the target at each location in the entire original image.

All of these features distinguish YOLOv3 from other detector models; however, accuracy and precision metrics are achieved when conducting a training process with a broad set of data that must contain multiple detections in the same image, which must be captured in different contexts and scales. In addition, the dataset can be diversified through the use of image augmentation techniques.

Finally, the analysis of the system in persuading people to wear masks is proposed for future work. The integration of additional sensors into the system to analyze air quality especially indoors is also considered.

Acknowledgments

The authors would like to thank the Private Technical University of Loja (UTPL) for funding this work through the second call «FUNDING DEGREE AND MASTER RESEARCH - 2020».

References

[1] R. Aragón Nogales, I. Vargas Almanza, and M. G. Miranda Novales, "COVID-19 por SARS-CoV-2: la nueva emergencia de salud," *Revista Mexicana de Pediatría*, vol. 86, pp. 213–218, 2020. [Online]. Available: https://dx.doi.org/10.35366/91871

[2] WHO, "Listings of WHO's response to COVID-19," World Health Organization. [Online]. Available: https://bit.ly/3mAZ6LH

[3] ——, "Vías de transmisión del virus de la COVID-19: Repercusiones para las recomendaciones relativas a las precauciones en materia de prevención y control de las infecciones." [Online]. Available: https://bit.ly/3epu4Sq

[4] OMS, "Who coronavirus (COVID-19) dashboard," 2021. [Online]. Available: https://bit.ly/3mDAO3r

[5] OPS, "Vacunas contra la COVID-19," 2020. [Online]. Available: https://bit.ly/3z0JGFs

[6] H. Ritchie, E. Mathieu, L. Rodés-Guirao, C. Appel, C. Giattino, E. Ortiz-Ospina, J. Hasell, B. Macdonald, D. Beltekian, M. Roser, and et al., "Coronavirus (COVID-19) vaccinations - statistics and research," 2020. [Online]. Available: https://bit.ly/3sEmtro

[7] C. Costa and C. Tombesi, "COVID-19: Cuánto tiempo se demoró en encontrar la vacuna para algunas enfermedades (y por qué este coronavirus es un caso histórico)," 2020. [Online]. Available: https://bbc.in/3pEV0Eh

[8] "Comparative research grant," *Anthropology News*, vol. 36, no. 8, pp. 43–43, 1995. [Online]. Available: https://anthrosource.onlinelibrary.wiley.com/doi/abs/10.1111/an.1995.36.8.43.1

[9] S. S. Bibak Sareshkeh, E. Magli, and P. Dal Zovo, "Combined ict technologies for supervision of complex operations in resilient communities," Master's thesis, 2020. [Online]. Available: https://bit.ly/3HaioPE

[10] I. Santos-González, A. Rivero-García, J. Molina-Gil, and P. Caballero-Gil, *Implementation and Analysis of Real-Time Streaming Protocols*, vol. 17, no. 4, 2017. [Online]. Available: https://doi.org/10.3390/s17040846

[11] A. Nurrohman and M. Abdurohman, "High performance streaming based on H264 and real time messaging protocol (RTMP)," in *2018 6th International Conference on Information and Communication Technology (ICoICT)*, 2018, pp. 174–177. [Online]. Available: https://doi.org/10.1109/ICoICT.2018.8528770

[12] S. Basu, "What are video streaming codecs & container formats: Muvi live server," 2020. [Online]. Available: https://bit.ly/3ErJPCZ

[13] J. S. Katz, "Aiot: Thoughts on artificial intelligence and the internet of things," *IEEE Internet if Things*, 2019. [Online]. Available: https://bit.ly/3sBwGEZ

[14] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *ArXiv*, vol. abs/1804.02767, 2018. [Online]. Available: https://bit.ly/3psJLyp

[15] A. M. Porcelli, "La inteligencia artificial y la robótica: sus dilemas sociales, éticos y jurídicos," *Derecho global. Estudios sobre derecho y justicia*, vol. 6, pp. 49–105, 2020. [Online]. Available: https://doi.org/10.32870/dgedj.v6i16.286

[16] X. Jiang, T. Gao, Z. Zhu, and Y. Zhao, "Real-time face mask detection method based on YOLOv3," *Electronics*, vol. 10, no. 7, p. 837, 2021. [Online]. Available: https://doi.org/10.3390/electronics10070837

[17] S. Sethi, M. Kathuria, and T. Kaushik, "Face mask detection using deep learning: An approach to reduce risk of coronavirus spread," *Journal of Biomedical Informatics*, vol. 120, p. 103848, 2021. [Online]. Available: https://doi.org/10.1016/j.jbi.2021.103848

[18] D. González Dondo, J. A. Redolfi, R. G. Araguás, and D. García, "Application of deep-learning methods to real time face mask detection," *IEEE Latin America Transactions*, vol. 19, no. 6, pp. 994–1001, 2021. [Online]. Available: https://bit.ly/3pw7DkM

[19] S. Sethi, M. Kathuria, and T. Kaushik, "A real-time integrated face mask detector to curtail spread of coronavirus," *Computer Modeling in Engineering & Sciences*, vol. 127, no. 2, pp. 389–409, 2021. [Online]. Available: https://doi.org/10.32604/cmes.2021.014478

[20] I. Vich, "Medical masks dataset images tfrecords," Kaggle, 2020. [Online]. Available: https://bit.ly/3er0tb8

[21] S. Ge, J. Li, Q. Ye, and Z. Luo, "MAFA," 2018. [Online]. Available: https://bit.ly/3FBC52o

[22] S. Yadav and S. Shukla, "Analysis of k-Fold Cross-validation over hold-out validation on colossal datasets for quality classification," in *2016 IEEE 6th International Conference on Advanced Computing (IACC)*, 2016, pp. 78–83. [Online]. Available: https://doi.org/10.1109/IACC.2016.25

[23] E. Allibhai, "Holdout vs. Cross-validation in machine learning." 2018. [Online]. Available: https://bit.ly/3z2TbE0

[24] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2021. [Online]. Available: https://doi.org/10.1109/JPROC.2020.3004555

[25] L. Herrera-Izquierdo and M. Grob, "A performance evaluation between docker container and virtual machines in cloud computing architectures," *Maskana*, vol. 8, pp. 127–133, 2017. [Online]. Available: https://bit.ly/3z12oNf

[26] NVIDIA, "Jetpack sdk 4.5.1 archive," 2021. [Online]. Available: https://bit.ly/32BxzT1

[27] Python, "Welcome to python.org," 2021. [Online]. Available: https://bit.ly/3qqTd4Q

[28] NVIDIA, "Quickstart guide - deepstream 6.0 release documentation," 2021. [Online]. Available: https://bit.ly/3sDTa8s

[29] ProminenceAI, "Prominenceai/deepstream-services-library: A shared library of on-demand deepstream pipeline services for Python and C/C++," GitHub. [Online]. Available: https://bit.ly/3pyxM2y

[30] MongoDB, "The application data platform," MongoDB. [Online]. Available: https://bit.ly/3qrRsUL

[31] N. Craig-Wood, "Rclone syncs your files to cloud storage," 2014. [Online]. Available: https://bit.ly/3JlPNsu

[32] Docker, "Empowering app development for developers," 2020. [Online]. Available: https://www.docker.com/

[33] A. Thakur, C. Clauss, C. Hollinger, V. Boivin, B. Lowe, M. Schoentgen, and R. Bouckenooghe, "abhiTronix/vidgear: VidGear v0.2.3," Oct. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5602375

[34] OpenCV. (2021) Opencv courses holiday sale. [Online]. Available: https://bit.ly/3ezvAS1

[35] Google Developers, "Firebase," 2020. [Online]. Available: https://bit.ly/3JinCeh

[36] Pallets, "Flask web development, one drop at a time," Pallet, 2010. [Online]. Available: https://bit.ly/3Hemy9h

[37] J. T. Mark Otto. (2021) Build fast, responsive sites with bootstrap. [Online]. Available: https://bit.ly/32Nl5rK

[38] Google. (2021) Colaboratory. Google Research. [Online]. Available: https://bit.ly/3EC3mk0

[39] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Springer International Publishing, 2014, pp. 740–755. [Online]. Available: https://bit.ly/3sxpZUu

[40] M. S. Aslanpour, S. S. Gill, and A. N. Toosi, "Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research." *Internet of Things*, vol. 12, p. 100273, 2020. [Online]. Available: https://doi.org/10.1016/j.iot.2020.100273

[41] M. Ashouri, F. Lorig, P. Davidsson, and R. Spalazzese, "Edge computing simulators for iot system design: An analysis of qualities and metrics," *Future Internet*, vol. 11, no. 11, p. 235, 2019. [Online]. Available: https://doi.org/10.3390/fi11110235

[42] F. Oliveira-Teixeira, T. P. Donadon-Homem, and A. Pereira-Junior, "Aplicación de inteligencia artificial para monitorear el uso de mascarillas de protección," *Revista Científica General José María Córdova*, vol. 19, no. 33, pp. 205–222, 2021. [Online]. Available: https://doi.org/10.21830/19006586.725